UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Funciones de adquisición heurísticas para Optimización Bayesiana

Autor: Luis Carlos Jariego Pérez
Tutor: Eduardo César Garrido Merchán

Junio 2019

Luis Carlos Jariego Pérez
Funciones de adquisición heurísticas para Optimización Bayesiana

Luis Carlos Jariego Pérez
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

’No te dejes abrumar si tu objetivo parece demasiado grande.
Céntrate en dar un pequeño paso cada vez.’

-Confucio

Agradecimientos

Me gustaría agradecer a todas las personas que han hecho posible la elaboración de este documento y todo el trabajo que hay detrás.

En particular, a mi tutor Eduardo Garrido por aportarme orientación.

A la Universidad Autónoma, y sus profesores, por proporcionar el marco necesario, y al Centro de Computación Científica por permitirme hacer uso de una cuenta para correr los experimentos.

Y a mi familia y amigos por su apoyo.

Resumen

La Optimización Bayesiana (OB) es un campo con una creciente popularidad actualmente debido a su utilización en Aprendizaje Automático, ya que permite la configuración de algoritmos, mediante el ajuste de hiperparámetros, sin la intervención humana. Sin embargo, esta es tan sólo una de sus numerosas aplicaciones.

Se trata de una herramienta utilizada para resolver problemas de optimización global sobre funciones objetivo de caja negra, esto es, que no tienen expresión analítica conocida, con evaluaciones muy costosas y cuyas observaciones pueden ser ruidosas. Para ello, hace uso de un modelo probabilístico (típicamente un Proceso Gaussiano) que representa la incertidumbre sobre el objetivo, y construye una función de adquisición de forma iterativa sobre el mismo, que son criterios de utilidad que deciden los puntos a evaluar en la búsqueda del óptimo.

Con ese fin, se intenta mantener un balance entre exploración y explotación del dominio. En la práctica, las funciones de adquisición favorecen una o la otra, y no existe una mejor opción para todos los problemas posibles. Este trabajo explora distintos enfoques para tratar de adaptar las funciones de adquisición de la OB a la mayor parte de los problemas de optimización.

Se presentarán heurísticas con el fin de construir funciones de adquisición que alternen o combinen de distintas formas los criterios disponibles en un portfolio. Se espera con ello obtener mejoras en el rendimiento sobre los distintos criterios estándar encontrados habitualmente en la literatura. Para comprobarlo, se harán pruebas experimentales con funciones tipo benchmark a optimizar y un escenario real de clasificación.

Palabras clave

Optimización bayesiana, función de adquisición

Abstract

Bayesian Optimization (BO) is a field with increasing popularity nowadays thanks to its use in automated Machine Learning, as it allows the configuration of algorithms, via the tuning of its hyperparameters, without human intervention. However, that is only one of its numerous applications.
BO is a powerful tool used to solve global optimization problems over black-box objective functions that do not have a known closed analytic expression, are very expensive to evaluate and whose observations may be corrupted by noise. It relies on a probabilistic model that represents the uncertainty over the objective (typically a Gaussian Process), and builds an acquisition function iteratively over it, which is a utility criterion that selects the points that will be evaluated in the search of the optimum.
To that end, they try to find a good trade-off between exploration and exploitation of the domain. In practice, they tend to favor one or the other and there is no best or obvious option for every possible problem. This work explores different approaches to try to adapt the acquisition functions of BO to most optimization problems.
We are going to present heuristics in order to build some acquisition functions based on switching or combining in different ways the available criteria in a portfolio. We expect to obtain improvements in the resulting performance compared to the different standard criteria usually found in literature. In order to test that, some experiments will be carried out with typical benchmark functions to optimize and a real scenario involving a classification problem.

Keywords

Bayesian optimization, acquisition function

Índice

1 Introducción **1**

1.1 Introducción 1

1.2 Descripción breve del problema 2

1.3 Organización del documento 3

2 Estudio del estado del arte **5**

2.1 Hiperparametrización de algoritmos de Aprendizaje Automático 5

2.2 Optimización Bayesiana 6

2.3 Modelos probabilísticos 7

2.4 Funciones de adquisición 9

2.5 Recapitulación 11

3 Definición del proyecto **13**

3.1 Objetivos 13

3.2 Asunciones 13

3.3 Hipótesis 14

3.4 Restricciones y alcance 14

4 Diseño y desarrollo **15**

4.1 Diseño del sistema 15

4.2 Funciones de adquisición propuestas 16

4.2.1 Selección aleatoria (*random*) 16

4.2.2 Secuencial (*sequential*) 17

4.2.3 Ponderada (*weighted*) 17

4.2.4 Ruidosa 18

4.3 Metaoptimización 19

5 Implementación **21**

5.1 Entorno utilizado 21

5.2 Procedimiento y detalles 21

6 Experimentos y resultados **23**

6.1 Descripción y diseño de los experimentos 23

6.2 Resultados de los experimentos 27

7 Conclusiones y trabajo futuro **31**

7.1 Objetivos y alcance 31

7.2 Impresiones y dificultades 31
7.3 Conclusiones 32
7.4 Trabajo futuro 32

Bibliografía **36**

Acrónimos **37**

Apéndices **39**

A Trabajo auxiliar **41**

B Plots **45**

Listas

Lista de algoritmos

2.1 Algoritmo GP-Hedge. 10

Lista de figuras

2.1 Proceso Gaussiano 11

4.1 Diagramas de flujo del sistema 15

5.1 Proceso de optimización 22

6.1 Función Branin 24
6.2 Función Rastrigin 25
6.3 Convergencia Branin 27
6.4 Convergencia Rastrigin 28
6.5 Convergencia Hartmann-3 29
6.6 Convergencia problema real 30

A.1 Abstract BISP 11 42
A.2 Poster BISP 11 43

B.1 Ejemplo plot optimización 45
B.2 Ejemplo plot optimización 2 46
B.3 Mínimos en la función Branin 46
B.4 Mínimo en la función Rastrigin 47
B.5 Convergencia de la función Schaffer nº2 47

Lista de tablas

6.1 Tabla de experimentos 26
6.2 Tabla de resultados 30

1

Introducción

1.1. Introducción

Supóngase que se entra a un casino con múltiples máquinas tragaperras y se debe decidir cuál elegir para ganar el máximo dinero posible en el menor número de tiradas. Este escenario se puede formalizar de la siguiente manera: se tiene un número finito de intentos $t = 1, 2, ..., T$ determinados por un tiempo máximo, presupuesto o recursos. En cada intento hay un conjunto finito de posibles acciones a tomar $A = 1, 2, ..., K$ y por cada una de ellas una recompensa $x(t)$.

En el ejemplo, T podría estar determinado por el dinero que se tiene o la hora a la que cierra el casino, K serían el número de máquinas disponibles, y la recompensa $x(t)$, el dinero o beneficio que da la máquina escogida en el instante t. Definimos arrepentimiento o *regret* como la diferencia entre la recompensa obtenida y la que daría la elección de la mejor acción en cada instante.

Este es el llamado *Multi-Armed Bandit Problem*, un problema clásico del aprendizaje por refuerzo (*reinforcement learning*). [1] Este campo, perteneciente al Aprendizaje Automático (AA) , tiene como objetivo determinar las acciones a tomar por un agente para maximizar la recompensa (o minimizar el arrepentimiento). Se trata de un tipo de problemas de optimización complicado, para el que los métodos habituales de resolución no son efectivos o eficientes, y uno de los numerosos problemas para los que la Optimización Bayesiana (OB) es un buen método de resolución.

La OB es una herramienta efectiva para la optimización de funciones black-box costosas de evaluar. Suele aplicarse por su potencia resolutiva en distintas áreas como el desarrollo de sitios web o interfaces de usuario interactivas (mediante test A/B), redes de sensores [2], robótica [3], diseño de experimentos [4] y automatic *Machine Learning (ML)* (esto es, ajuste de hiperparámetros en algoritmos de AA), entre otros. Un ejemplo reciente de esto último es su uso en el famoso programa *AlphaGo* de Google. [5]

El tema a tratar en este proyecto será la optimización global mediante el uso de la Optimización Bayesiana. Para ello, es útil extrapolar el problema del *Multi-Armed Bandit* a funciones cualesquiera, donde las acciones posibles serían la evaluación de cada punto del espacio de búsqueda, por tanto

puede haber infinitos, y la recompensa es el objetivo real, esto es, el valor de la función en el punto seleccionado.

Un aspecto clave de la OB son las funciones de adquisición, que indican los puntos de la función que se deben evaluar en busca del óptimo otorgándoles un mayor valor. Cuando se dispone de pocas evaluaciones de la función objetivo, la elección de una buena función de adquisición puede cambiar significativamente el resultado de la optimización.

Ya en el escenario anterior se encuentra el también clásico *dilema de compensación entre exploración y explotación*: [1] ¿es mejor quedarse en la máquina en la que se está o moverse a otra? En definitiva, se trata del balance entre explotar la opción que ha dado las mayores recompensas en el pasado o explorar nuevas opciones que podrían (o no) dar mayores recompensas en el futuro. Este será un aspecto recurrente, pues las funciones de adquisición suelen favorecer una opción o la otra.

La motivación del presente proyecto es el estudio de la Optimización Bayesiana y su aplicación al ajuste de hiperparámetros en Aprendizaje Automático. Cuando se emplea con este propósito, se desea despreocuparse de la elección de parámetros para cierto algoritmo. Sin embargo, para la utilización de OB hay que elegir la función de adquisición a emplear (entre otras cosas), lo que se convierte en otro problema por sí mismo, dado que esto determina en muchos casos el resultado de la optimización. Se desea proponer nuevas funciones de adquisición, con distintos enfoques, que puedan funcionar o se adapten a la mayor parte de los problemas y permitan, por tanto, una mayor automatización del proceso.

1.2. Descripción breve del problema

Como ya se ha indicado, nos vamos a centrar en la optimización global de funciones. Por tanto, estamos interesados en resolver el problema matemático de encontrar un mínimo para una función objetivo f:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\chi} f(\mathbf{x}), \tag{1.1}$$

donde $\chi \subseteq \mathbb{R}^d$ compacto es el espacio de búsqueda deseado. El problema de encontrar el máximo de una función f puede reformularse como el mínimo de $-f(x)$. Por ello, a partir de ahora hablaremos sólo de minimización.

La OB se dedica a la optimización de funciones bajo las siguientes restricciones:

- f es una función de caja negra (*black-box*), sin expresión analítica cerrada conocida, y gradientes desconocidos.
- f es una función muy costosa de evaluar, bien en tiempo o recursos.
- Las observaciones resultantes de evaluar f pueden ser ruidosas.

Un ejemplo ilustrativo de este tipo de problemas sería, por ejemplo, el establecimiento de un pozo petrolífero: se desea colocar una bomba que extraiga combustible de forma óptima, pero se desconoce la localización exacta donde se debe realizar para encontrar el crudo. Además, cada perforación resulta de gran coste para la compañía.

Se puede comprobar ahora que los problemas mencionados en el apartado anterior se ajustan a esta descripción.

Para funciones objetivo sin las restricciones enunciadas anteriormente, otros tipos de optimización serán mejores en términos de eficiencia (cálculo diferencial, programación lineal, etc). Por ejemplo, si la función no es cara de evaluar pero su gradiente sí es desconocido, una buena solución es la utilización de una metaheurística, como por ejemplo un algoritmo genético. [6]

1.3. Organización del documento

La estructura del documento va a seguir los siguientes apartados:

- Estudio del estado del arte. Constituirá un repaso desde los aspectos más básicos de la Optimización Bayesiana hasta los últimos avances en el área.
- Definición del proyecto. En este apartado se discutirán los objetivos, asunciones previas, hipótesis, restricciones y alcance del proyecto.
- Diseño y desarrollo, donde se ahondará en el diseño y modelo y diseño seguidos para la realización del proyecto.
- Implementación. Aquí se verá en más detalle cómo y con qué herramientas se ha llevado a cabo.
- Experimentos y resultados. Se mostrarán e interpretarán los resultados de las pruebas realizadas.
- Conclusiones y trabajo futuro. Se expondrá una reflexión final y se propondrá una continuación al proyecto.
- Bibliografía. Se listarán las citas y referencias.
- Acrónimos.
- Apéndices, donde se incluirán figuras, resultados y otros recursos que no tengan cabida en el resto del documento.

2

Estudio del estado del arte

En este apartado se va a indagar y recopilar el estado de la Optimización Bayesiana en la actualidad de forma concisa. Se empezará por una breve introducción con los métodos habituales para el ajuste de hiperparámetros en Aprendizaje Automático para continuar con la idea general de la OB y sus principales componentes: modelos probabilísticos y funciones de adquisición.

2.1. Hiperparametrización de algoritmos de Aprendizaje Automático

Actualmente encontramos el Aprendizaje Automático aplicado a numerosos y diferentes ámbitos, donde se pretenden automatizar tareas y realizar predicciones. Por ello, han surgido gran cantidad de algoritmos con muchas posibilidades, que traen consigo un problema común: el ajuste de sus hiperparámetros.

Normalmente no se sabe, por ejemplo, cual es la profundidad de los árboles de decisión en un random forest o el número de neuronas en las capas ocultas de una red neuronal óptimas para un problema concreto. Es por eso que la selección de estos hiperparámetros mediante optimización está a la orden del día.

Hay diferentes métodos habituales utilizados con este fin:

- Consulta a expertos, que hacen uso de conocimiento previo del problema. No es útil para problemas nuevos, distintos a los ya conocidos.
- Búsqueda de rejilla (grid search): se divide el espacio de búsqueda en una 'cuadrícula' de puntos a evaluar. Para problemas de múltiples dimensiones se trata de una búsqueda muy costosa.
- Búsqueda aleatoria, mediante la evaluación de puntos seleccionados al azar. Es mejor que la búsqueda de rejilla en algunos sentidos pero sigue siendo costosa si se quiere garantizar una buena cobertura del espacio.
- Uso de metaheurísticas como los algoritmos evolutivos. No son muy eficientes, sobre todo si las evaluaciones son costosas.
- Optimización Bayesiana, que para este tipo de problemas resulta un método especialmente efectivo. [7]

Ninguno de estos métodos garantiza la convergencia al óptimo global del problema, pero pueden

compararse en términos de eficiencia: qué mínimo encuentran en función al tiempo o número de evaluaciones realizadas.

2.2. Optimización Bayesiana

La idea principal de la Optimización Bayesiana es la construcción de un modelo probabilístico subrogado de forma secuencial para tratar de inferir la función objetivo. De forma iterativa se realizan nuevas observaciones y se va actualizando el modelo, reduciendo la incertidumbre del mismo. Esto permite trabajar con un modelo conocido y más barato, que se usa para construir una función de utilidad que determine el siguiente punto a evaluar. A continuación se describen los distintos pasos de la metodología de la OB.

En primer lugar hay que elegir un *modelo a priori* sobre el posible espacio de funciones, para ello pueden usarse distintos enfoques paramétricos, como Beta-Bernoulli Bandit o Modelos Lineales (Generalizados), o bien modelos no paramétricos como los Procesos t-Student o los Procesos Gaussianos. [8]

Después, de forma repetida hasta cierto criterio de parada:

Se combina el prior y la verosimilitud de las observaciones hasta el momento para obtener una distribución a posteriori. Esto se hace mediante el teorema de Bayes, de ahí el origen del nombre.

Recordemos el teorema de Bayes. Sean A y B dos sucesos tales que se conoce la probabilidad condicionada $P(B|A)$, entonces la probabilidad $P(A|B)$ viene dada por:

$$\mathrm{P}(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{2.1}$$

donde $P(A)$ es la probabilidad *a priori*, $P(B|A)$ es la probabilidad del suceso B condicionada a la ocurrencia del suceso A, y $P(A|B)$ es la probabilidad *a posteriori*.

Entonces, se maximiza cierta función de utilidad sobre el modelo a posteriori para determinar el siguiente punto a evaluar. Para terminar, se evalúa la función en dicho punto y se recopila la nueva observación para volver a repetir hasta el criterio de parada.

Existen otras aplicaciones de la Optimización Bayesiana, o variaciones del problema, como la optimización de funciones multiobjetivo, dominios con restricciones, OB el altas dimensiones, etc.

2.3. Modelos probabilísticos

En Optimización Bayesiana se utilizan modelos probabilísticos de regresión, que pueden ser utilizados para definir una distribución sobre funciones, por ello resultan especialmente útiles para el problema. De forma intuitiva, estos modelos son una creencia de cómo es o 'por dónde' puede encontrarse nuestra función objetivo. Es una forma de cuantificar y representar la incertidumbre sobre la misma. A medida que se hacen nuevas observaciones (evaluaciones de la función), se ajusta el modelo, de forma que lo 'hacemos pasar' por ese punto. Como se puede deducir, esto nos servirá a la hora de decidir qué puntos evaluar para encontrar el óptimo, pues tendremos una idea de en qué zonas tiene una mayor probabilidad de encontrarse.

El modelo a priori más empleado en OB es el Proceso Gaussiano (GP) .

La literatura define Proceso Gaussiano como una "*colección de variables aleatorias, tal que cualquier número finito de ellas tienen distribuciones Gaussianas multivariantes (consistentes)*." [9]

La distribución normal (o Gaussiana) multivariante, que se define sobre vectores aleatorios, está completamente determinada por un vector de medias y una matriz de covarianzas. Un GP es una generalización de la distribución normal sobre funciones, y está completamente especificado por una función de medias $\mu(x)$ y una función de covarianzas $k(x, x')$. Se trata de una función de densidad de probabilidad de infinitas dimensiones que asigna a cada punto x del dominio una distribución de probabilidad normal para $f(x)$, el valor de la función aleatoria f.

El modelo utilizado son funciones distribuídas como Procesos Gaussianos:

$$f \sim GP(\mu(x), k(x, x') \tag{2.2}$$

Dado un conjunto de datos D, la función de medias y la de covarianzas a posteriori $\mu(x|D)$ y $k(x|D)$ pueden ser calculadas de forma explícita mediante el teorema de Bayes.

La capacidad del proceso Gaussiano de expresar una distribución de funciones adecuada depende tan sólo de la función de covarianzas k, a menudo llamada *kernel*, que puede contener suposiciones fuertes sobre las funciones más probables. [7] Los más comúnmente utilizados son los llamados *Mátern kernels*, que son bastante flexibles. Están parametrizados por un parámetro de suavizado $\nu > 0$: las funciones resultantes con dicho kernel son diferenciables $\lfloor \nu - 1 \rfloor$ veces.

El *squared exponential* (*SE*) kernel, también bastante utilizado, es un caso especial de Mátern cuando $\nu \to \infty$, por lo que suele resultar en funciones con un suavizado poco realista para algunos tipo de problemas, como el que nos incumbe. Su expresión es:

$$K_{SE}(\mathbf{x}, \mathbf{x}') = \theta_0 exp\{-\frac{1}{2} r^2(\mathbf{x}, \mathbf{x}')\} \tag{2.3}$$

$$r^2(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D} \frac{(x_d - x'_d)}{\theta_d^2}$$

La elección de una función de covarianzas apropiada y de los hiperparámetros asociados es poco clara para problemas prácticos. Es algo que quedará fuera de este estudio.

El coste computacional de la inferencia exacta en GPs es $O(n^3)$, donde n es el número de observaciones, esto es, crece de forma cúbica con n. Existen alternativas que reducen el tiempo de cómputo cuando se trabaja con gran cantidad de datos, como los Sparse Gaussian Processes (SPGPs, SSGPs), que son variaciones sobre GPs que no realizan una inferencia exacta; o el uso de regresión mediante random forests, que emplean árboles de decisión sobre subconjuntos aleatorios de las observaciones, reduciendo así el coste. [8]

Hay otros modelos posibles que funcionan bien como modelos probabilísticos para Optimización Bayesiana y tratan de dar solución a problemas como la falta de flexibilidad de los GP: otros modelos no paramétricos, como los Procesos t-Student [10], un modelo derivado de los GP y Procesos Inversos de Wishart; o paramétricos como las Bayesian (Deep) Neural Network [11], que atacan también su falta de escalabilidad.

2.4. Funciones de adquisición

Como ya se ha mencionado, es necesario definir también una función de utilidad para recabar nuevos datos de acuerdo a cierto criterio de optimalidad, como lo son las funciones de adquisición. Estas hacen uso del estado del modelo probabilístico elegido para tratar de maximizar la ganancia o minimizar la pérdida. En cierta manera, se convierte el problema de optimización en uno de decisión. Hay muchas y variadas alternativas para elegir. La clave suele estar en hacer un buen balance entre exploración y explotación.

A continuación, se va a hacer un repaso a las funciones de adquisición más mencionadas en la literatura de la Optimización Bayesiana (con GPs). Se hará uso de su nombre y siglas en inglés.

Probability of Improvement

Es una estrategia intuitiva, y la que primero aparece en la literatura (1964), que consiste en maximizar la *probabilidad de mejora* sobre el mejor valor obtenido hasta el momento $f(\mathbf{x}_{best})$. Con GPs, esta función se puede computar analíticamente como:

$$\alpha_{\mathrm{PI}}(\mathbf{x}) = \mathbb{P}[f(\mathbf{x}) > f(\mathbf{x}_{best})] = \Phi(\gamma(\mathbf{x})), \tag{2.4}$$

donde

$$\gamma(\mathbf{x}) = \frac{\mu(\mathbf{x}) - f(\mathbf{x}_{best})}{\sigma(\mathbf{x})}, \tag{2.5}$$

y Φ es la función de distribución acumulada (CDF) de una distribución normal estándar.

Expected Improvement

Alternativamente, se podría elegir maximizar la *mejora esperada*, que incorporaría la 'cantidad' de mejora sobre el óptimo actual. También tiene forma cerrada bajo GPs:

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = \sigma(\mathbf{x})(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \phi(\gamma(\mathbf{x}))), \tag{2.6}$$

donde ϕ es la función de densidad (PDF) de una distriución normal estándar y γ definida en 2.5.

Es probablemente la adquisición más usada. En algunos problemas es demasiado codiciosa, es decir, favorece la explotación, por eso a veces se hace trata de hacerla más explorativa mediante la inclusión de un parámetro ψ como sumando en $\gamma(\mathbf{x})$.

GP Lower Confidence Bound

Un desarrollo más reciente es la idea de explotar los *límites inferiores de confianza* (*lower*) del GP, o superiores (*upper*) si se desea maximizar, para construir funciones de adquisición que minimizan el regret durante la optimización. [12] Esta función de adquisición viene dada por la siguiente expresión:

$$\alpha_{\mathrm{LCB}}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x}), \tag{2.7}$$

con un parámetro $\kappa \geq 0$ para balancear la explotación (primer término) frente a la exploración (segundo término). La formalización con el regret puede tener más sentido para ciertas configuraciones. Los autores definen el *instantaneous regret* como $r(\mathbf{x}) = f(\mathbf{x}^*) - f(\mathbf{x})$.

GP Hedge

GP-Hedge es un algoritmo más actual (2011) que consiste en, a partir de N funciones de adquisición, seleccionar los puntos 'nominados' por cada una de ellas y elegir cada función de adquisición con probabilidad proporcional a la ganancia generada por la misma. Pertenece a las llamadas extrategias portfolio. [13] Se pasa a describir del algoritmo:

Seleccionar el parámetro $\eta \in \mathbb{R}$;
Establecer $g_0^i = 0$ para $i = 1, ..., N$;
for $i \leftarrow 1$ **to** N **do**
    Nominar puntos de cada función de adquisición: $\mathbf{x}_t^i = argmax_x u_i(\mathbf{x}|\mathcal{D}_{1:t-1})$;
    Seleccionar el nominado $\mathbf{x}_t = \mathbf{x}_t^j$ con probabilidad $p_t(j) = \exp(\eta g_{t-1}^j) / \sum_{l=1}^k \exp(\eta g_{t-1}^l)$;
    Evaluar la función objetivo $y_t = f(\mathbf{x}_t) + \epsilon_t$;
    Aumentar los datos $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$;
    Recibir recompensas $r_t^i = \mu_t(\mathbf{x}_t^i)$ del GP actualizado;
    Actualizar ganancias $g_t^i = g_{t-1}^i + r_t^i$;
end

Algoritmo 2.1: Algoritmo GP-Hedge.

Otros criterios

Existen otras funciones de adquisición, como las políticas basadas en la información (*information-based policies*) que, en contraste con las anteriores, consideran la distribución a posteriori sobre el minimizador desconocido $\mathbf{x}^*$, denotada $p_*(\mathbf{x}|D_n)$, inducida de forma implícita por la posterior sobre funciones objetivo f. Las principales son Thompson Sampling (TS), que toma como función de recompensa muestras aleatorias de la posterior: $\mathbf{x}_{n+1} \sim p_*(\mathbf{x}|D_n)$; y Entropy Search (ES) y sus variantes PES y MES, que seleccionan como siguiente punto el que se espera que cause la mayor reducción de entropía de la distribución $p_*(\mathbf{x}|D_n)$. [8]

También existen algoritmos que utilizan información sobre derivadas (del modelo probabilístico)

para encontrar buenas soluciones con un menor número de evaluaciones, como el Knowledge Gradient. [14] Tiene una versión que permite realizar OB por lotes (Parallel Knowledge Gradient).

2.5. Recapitulación

Hemos visto que la OB por tanto, es una estrategia que convierte el problema de optimización global en una serie de problemas de la forma:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\chi} \alpha(\mathbf{x}) \tag{2.8}$$

Es decir, a cada paso se maximiza la utilidad α, poco costosa de evaluar y con gradientes conocidos normalmente. En la práctica, se suele recurrir a la discretización de la función o a un optimizador auxiliar.

En la figura 2.1 se muestra cómo se actualiza un modelo (GP en este caso), según nuevas observaciones de la función objetivo, que se realizan según determina la función de adquisición.

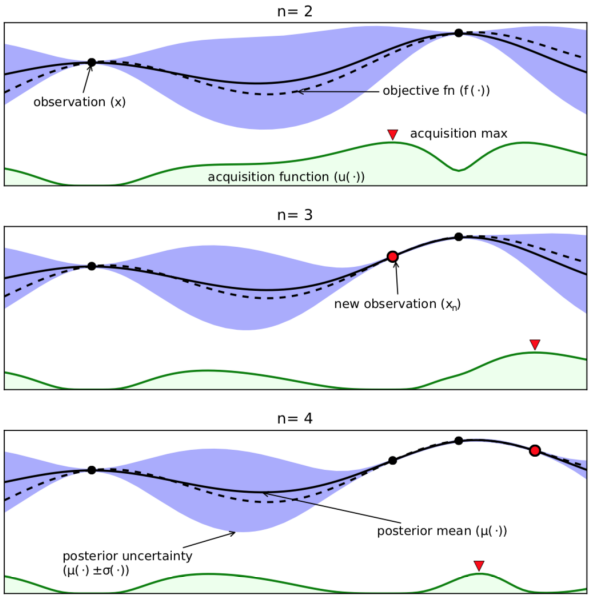


Figura 2.1: Ejemplo de actualización de un Proceso Gaussiano [8]

Las funciones de adquisición más tradicionales tienen carencias en cuanto a favorecer bien la exploración o la explotación en la búsqueda del óptimo, y serán buenas solo para cierto tipo de problemas concretos que lo requieran. Ninguna de ellas de forma individual ofrecerá un mejor desempeño sobre cualquier problema, y muchas veces no está claro cual se debe usar. Además, se ha observado de forma empírica que la mejor estrategia puede ir cambiando a lo largo de las distintas iteraciones de la optimización. [13]

3

Definición del proyecto

3.1. Objetivos

Con este proyecto se aportan ideas que contribuirán a la automatización de la OB mediante la proposición de nuevas funciones de adquisición heurísticas. Para ello nos basamos en la idea subyacente del 'No Free Lunch Theorem' [15] (AA y optimización), que se puede encontrar enunciado de distintas maneras:

- 'Dos algoritmos de optimización cualesquiera son equivalentes cuando su rendimiento se promedia sobre todos los posibles problemas.'
- 'Si un algoritmo de comporta bien en cierta clase de problemas, entonces necesariamente paga por ello con un rendimiento degradado en los problemas restantes.'

En definitiva, dice que ningún criterio de optimización es bueno para todos los posibles problemas.

Por tanto, se quieren construir funciones de adquisición con distintos enfoques que alternen o combinen distintos criterios disponibles en un portfolio, y que sean válidas para la mayoría de los problemas. Posteriormente se comprobará experimentalmente su comportamiento.

Los objetivos establecidos para este Trabajo de Fin de Grado son:

O-1.– Propuesta de nuevos modelos de funciones de adquisición.

O-2.– Implementación de los modelos descritos.

O-3.– Realización de experimentos e interpretación de los resultados.

O-4.– Comparación con funciones de adquisición que forman parte del estado del arte.

3.2. Asunciones

Previo al diseño y desarrollo del proyecto, se deben listar una serie de aspectos a tener en cuenta:

A-1.- Se asumirá que todas las funciones objetivo utilizadas de aquí en adelante se ajustan al escenario recomendable para aplicar la OB, y pertenecen al espacio de funciones definido por los priores de nuestro modelo probabilístico.

A-2.- Se supondrá que ciertas características observadas en las pruebas realizadas sobre estas funciones, son

extrapolables a otros problemas.

A-3.- También se ha supuesto que es posible e interesante trasladar las ideas del *ensemble learning* [16] o *stacking*, perteneciente al ámbito del Aprendizaje Automático, a funciones de adquisición para Optimización Bayesiana. La idea es conformar un portfolio con distintos modelos de funciones de adquisición para el mismo problema, de forma cada una de ellas puede ser buena para cierta parte del problema, pero no el problema entero. Estas funciones de adquisición se utilizarán como paso intermedio, y se combinarán o ponderarán mediante un meta-criterio para ofrecer un resultado único.
En AA, se suele utilizar con modelos con rendimiento parecido (*weak learners*), de forma que se equilibren los 'puntos flacos' de cada uno, sobre todo cuando no se conoce un modelo óptimo (o mejor que el resto) para un problema específico. Habitualmente, se obtiene así un modelo mejor y más robusto a cualquiera de los intermedios individuales que lo componen.

A-4.- Se considerará mejor en términos generales una función de adquisición para la que se converja en menor número de iteraciones a un óptimo menor que con otros criterios.

3.3. Hipótesis

El diseño de las nuevas funciones de adquisición se apoyará en algunas hipótesis:

H-1.- Combinando distintos criterios de utilidad se construirá uno que, aunque no rinda mejor en todos los problemas, no salga penalizado de forma significativa en la mayoría de ellos, encontrándose entre las mejores opciones.

H-2.- Mediante la alternación de distintos criterios en cada iteración de la OB se obtendrá un proceso de optimización que contenga los mejores rasgos de cada una de ellas.

H-3.- Cuanto mayor y más variado sea el portfolio de funciones empleado, mejor será el resultado del enfoque. Sin embargo, no hará falta un gran número de éstas para ir observando los efectos.

3.4. Restricciones y alcance

También debemos tratar de enmarcar y limitar el alcance de este proyecto mediante una serie de restricciones que listo a continuación.

R-1.- Solo se tratará con GPs como modelos a priori, y funciones uniobjetivo a minimizar.

R-2.- Las funciones de adquisición que se propondrán serán meta-criterios heurísticos sobre un portfolio de funciones mediante enfoques sencillos o *naive*.

R-3.- La implementación quedará enmarcada en las limitaciones de la librería sobre la que se realicen las adiciones o modificaciones.

R-4.- Los experimentos consistirán en la comparación de la optimización con diversas funciones de adquisición en varias funciones test estándar comunes en la literatura de la OB y un problema 'real' de ajuste de hiperparámetros en un algoritmo de AA.

R-5.- Para comparar el desempeño de las distintas funciones de adquisición se utilizará la convergencia al mínimo con respecto al número de iteraciones (y por tanto, de evaluaciones) realizado.

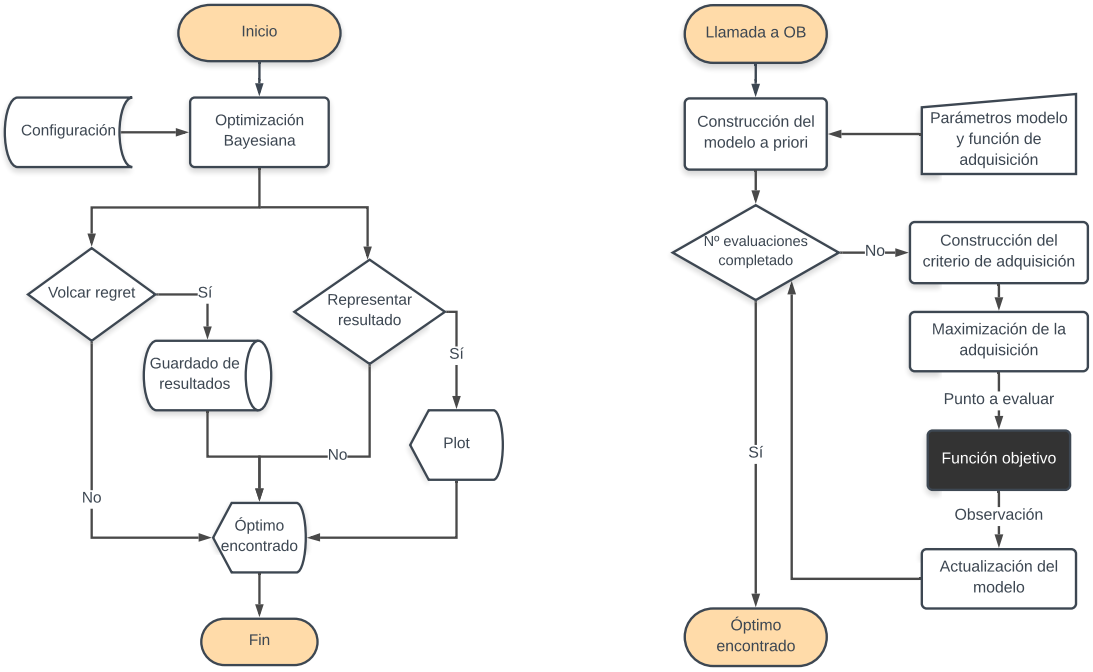
4

Diseño y desarrollo

Todo lo anterior conduce a la parte central de este proyecto, que es la proposición de nuevas funciones de adquisición para Optimización Bayesiana. Antes que nada se incluirá un breve apartado para indicar en qué parte del proceso afectará.

4.1. Diseño del sistema

En las figuras 4.1 se muestran diagramas de los dos procesos principales que describen el sistema en términos de Ingeniería del Software.



(a) Llamada al algoritmo de Optimización Bayesiana.

(b) Pasos principales del proceso de optimización.

Figura 4.1: Diagramas de flujo de los principales procesos del sistema.

A la izquierda se muestra un primer nivel con la llamada a un algoritmo de OB mediante un fichero de configuración, y permite volcar resultados o representar el resultado de la optimización. A la derecha se amplía el proceso de optimización describiendo los pasos principales y el flujo de datos en el mismo.

En los apartados siguientes se expondrán criterios de utilidad, que quedarán enmarcados dentro del proceso de construcción de la función de adquisición (4.1(b)).

4.2. Funciones de adquisición propuestas

En las funciones propuestas se hará uso de un portfolio o *set* de funciones de adquisición que se alternarán o combinarán de alguna forma. De aquí en adelante se considera el conjunto finito C compuesto por no menos de dos funciones de adquisición indexadas α_i, con $i \in I \subset \mathbb{N}$, conjunto de índices de las mismas.

$$C = \{\alpha_i\}_{i \in I} \tag{4.1}$$

Definiremos $N = card(C)$, el cardinal del conjunto o número de funciones del set, tal que $I = \{1, 2, ...N\}$.

A continuación, se pasa a describirlas.

4.2.1. Selección aleatoria (*random*)

Es una función de adquisición que selecciona de forma aleatoria en cada paso de la optimización entre las candidatas $\alpha_i \in S$. Formalmente:

$$R(\mathbf{x}) \leftarrow \mathrm{U}\{\alpha_i(\mathbf{x})\}_{i \in I} \tag{4.2}$$

Esto es, se elige al azar un $j \in I$, según una distribución uniforme discreta $\mathrm{U}(N)$ de tal forma que, en la iteración actual, $R(\mathbf{x}) = \alpha_j(\mathbf{x})$.

Se trata de una idea ingenua, que aleatoriza la función de adquisición seleccionada. Sin eambargo, existen ejemplos de funciones valle en las que algunos criterios muy avanzados fallan y uno aleatorio funciona. Por ejemplo, la función de Easom, donde el mínimo global tiene un pequeño área en relación al espacio de búsqueda, y el resto es un plano. La mayoría de los modelos no son capaces de lidiar con esto y una estrategia aleatoria puede ser la más indicada para conseguir que se evalúe un punto próximo al mínimo. [17]

4.2.2. Secuencial (*sequential*)

De forma parecida al anterior, este criterio selecciona la función de adquisición de manera secuencial, de forma que a cada paso se utiliza una de las candidatas:

$$S_t(\mathbf{x}) = \alpha_{tmod(N)}(\mathbf{x}), \tag{4.3}$$

donde t es la iteración actual, y S_t la adquisición en dicha iteración.

Se va eligiendo una función de adquisición distinta en cada paso de la optimización, independientemente de su rendimiento, lo cual puede ser útil al no seguir una única estrategia con la que puede que no se mejore o el proceso de optimización se quede 'estancado', como suele ocurrir, por ejemplo, con aquellas que favorecen la explotación.

Al ir alternando criterios, y si se elige un buen conjunto C, se deberían cancelar las tendencias de las mismas.

4.2.3. Ponderada (*weighted*)

Se trata de una función que devuelve una combinación afín de las funciones del set C:

$$W(\mathbf{x}) = \sum_{i \in I} \kappa_i \alpha_i(\mathbf{x}), \tag{4.4}$$

donde cada $\kappa_i \in [0, 1]$ es una constante a determinar tal que $\sum_{i \in I} \kappa_i = 1$ (afín).

Se ponderan mediante los pesos κ_i los valores obtenidos con las funciones de adquisición que conforman el portfolio, como si determinaran la importancia o contribución de cada una.

Eligiendo estos pesos de forma correcta puede ser una función de adquisición interesante que otorgue valores altos a puntos que todas las funciones candidatas valoran como posible punto a evaluar, pero que equilibre resultados donde unas funciones otorguen valores muy altos y otras no, y viceversa.

Los pesos pueden ser un hiperparámetro en forma de vector $\boldsymbol{w}$ tal que $\boldsymbol{w}_i = \kappa_i \in \mathbb{R}_+$, de forma que se pueda normalizar para que $\sum_{i \in I} \kappa_i = 1$, y por defecto, $\kappa_i = \frac{1}{N}$ para cada α_i.

Una opción de ampliación sería la adaptación de los pesos de la combinación en la misma función de acuerdo a algún criterio que hiciera uso de una recopilación de las mejoras obtenidas con cada función de adquisición del set, número de iteraciones sin mejorar o restantes, etc. al estilo de GP-Hedge (2.4).

4.2.4. Ruidosa

También se propone la posibilidad de agregar ruido aleatorio a cualquier función de adquisición. Mediante el parámetro κ se indica el nivel del ruido a añadir, generado según una distribución normal estándar.

$$N(\mathbf{x}) = \alpha(\mathbf{x}) + \kappa\epsilon, \tag{4.5}$$

siendo $\alpha(\mathbf{x})$ una función de adquisición cualquiera, la constante $\kappa > 0$ y $\epsilon \sim \mathcal{N}(0, 1)$. Esto es, el resultado de la función de adquisición seleccionada más un ruido aleatorio (controlado por una constante) en cada punto a evaluar.

La idea por la que esta opción podría ser interesante es añadir perturbaciones en la adquisición que puedan permitirnos evaluar puntos que de otra forma no se seleccionarían por haber cierta tendencia a la exploración o a la explotación.

Otra forma de añadir ruido a las funciones propuestas sería la inclusión en el conjunto C de una función de adquisición que fuera ruido aleatorio (Gaussiano, por ejemplo):

$$\alpha_{AL}(\mathbf{x}) \sim \mathcal{N}(0, 1) \tag{4.6}$$

4.3. Metaoptimización

La *metaoptimización* es el uso de cierto método de optimización para poner a punto (típicamente, ajustar hiperparámetros) otro, o el mismo, método de optimización.

En este caso, surgió la idea de realizar una optimización sobre los parámetros de las funciones de adquisición, que son parte del proceso de optimización de la OB, con el fin de automatizar el proceso. Se trata de un problema que se adapta perfectamente a las condiciones típicas de una función objetivo a optimizar por la OB: expresión analítica desconocida y evaluaciones costosas que pueden devolver resultados ruidosos. Por tanto, se decidió aplicar este método.

Este apartado se centra exclusivamente en la aplicación de la metaoptimización para la elección de hiperparámetros de la función de adquisición ponderada, es decir, de los pesos asociados a cada función de un set C fijado. Podría aplicarse de forma análoga a los hiperparámetros de otras adquisiciones.

Sea f una función objetivo, χ nuestro espacio de búsqueda, α la función de adquisición elegida y T el número de evaluaciones disponibles para f. Supongamos que se dispone de un algoritmo de Optimización Bayesiana, de la forma

$$OB(f, \chi, \alpha, T),$$

que devuelve el mínimo de la función encontrado en T iteraciones ($\mathbf{y}^T$), y el punto asociado donde se alcanza ($\mathbf{x}^T = \arg(\mathbf{y}^T)$).

Se procede bajo la suposición de que no existe un mismo óptimo para distintas funciones objetivo, así que la optimización estará sujeta a una función objetivo concreta f.

Por tanto, se tiene una función $f : \chi \to \mathbb{R}$ a optimizar, donde $\chi \subseteq \mathbb{R}^p$ es compacto, $p \in \mathbb{Z}^+$ el número de dimensiones del espacio de búsqueda; y la función de adquisición ponderada W, descrita en 4.2.3, con hiperparámetros los pesos asociados a cada función de adquisición del set, en forma de vector $\boldsymbol{w}$. Entonces, si se considera el mínimo real de f, $\mathbf{y}^*$ y un $T \in \mathbb{N}$ fijo, se puede definir una nueva función objetivo:

$\boldsymbol{F(\mathbf{x})}$ **:**

1: $\mathbf{y}^T = OB(f, \chi, W(\boldsymbol{w} = \mathbf{x}), T)$

2: devolver $|\mathbf{y}^T - \mathbf{y}^*|$

que devuelve el error absoluto o regret de la OB. F será la función objetivo de la metaoptimización.

En primer lugar, hay que establecer un espacio de búsqueda para los pesos, que son escalares que determinan la contribución de la función de adquisición asociada. Por tanto, se va a fijar que cada uno deba encontrarse en un intervalo $[0, c]$, con c $\in \mathbb{R}$ constante. Así, si se dispone de n funciones de adquisición, se establecerá como espacio de búsqueda un hipercubo $[0, c]^n$.

A continuación, se debe elegir una función de adquisición $\alpha : \chi \to \mathbb{R}$ con la que llevar a cabo la optimización. De esta forma, tan sólo habría que ejecutar

$$\mathbf{x}^T = \arg(OB(F, [0, c]^n, \alpha, T))$$

para obtener el punto mínimo $\mathbf{x}^T$ con los pesos que resultan en un menor regret para la optimización de la función f con función de adquisición ponderada.

5

Implementación

El objetivo de este apartado es describir las implementaciones llevadas a cabo, de manera que las pruebas, resultados y conclusiones de los siguientes apartados sean replicables en la medida de lo posible.

5.1. Entorno utilizado

A continuación, se detallan las características del entorno donde se ha llevado a cabo la implementación y ejecución de los experimentos.

- Se ha utilizado el lenguaje de programación *python*, por su versatilidad y sencillez, y disponer de una librería de optimización en la que estaban desarrollados todos los métodos necesarios para nuestras implementaciones. Se ha elegido la versión 3.4.10 para la ejecución de los experimentos por ser la más reciente disponible en el CCC.
- La librería sobre la que se ha trabajado y realizado modificaciones es *scikit-optimize* o *skopt* en su versión 0.5.2. La definen como una librería simple y eficiente que implementa métodos para optimización basada en modelos secuenciales. Se ha elegido esta librería entre otras por la sencillez, buena documentación y la 'reputación' que otorga el nombre de *scikit*.
- El código se encuentra disponible en un 'fork' realizado sobre la rama principal en mi página de *github*:
  `https://github.com/luisjariego/scikit-optimize`
  Por ese motivo no se ha adjuntado ninguna implementación en esta memoria.
- Para la generación de gráficos se ha utilizado *matplotlib* 3.1.0 para python, y *ggplot2* 3.1.1 en R, versión 3.4.4.

5.2. Procedimiento y detalles

El primer paso, previo al desarrollo del código, fue la familiarización con la librería de *skopt*, ver su funcionamiento y las funciones de adquisición disponibles. Contamos con todas las descritas en el apartado 2.4: PI, EI, LCB y GP-Hedge.

En segundo lugar, se escribió un programa que automatiza la ejecución de OB sobre distintas funciones objetivo. Para ello, se hizo que extrajera toda la información necesaria de un archivo JSON con el formato correcto, donde se incluyen la función objetivo y las de adquisición a utilizar junto con

todos los hiperparámetros de la OB. Esto facilitaría más adelante la ejecución de gran número de experimentos de forma sencilla. También se facilitó la visualización tanto de las funciones objetivo (en dos o tres dimensiones), así como del proceso de optimización y el ajuste del GP para funciones en dos dimensiones.

Para finalizar, se implementaron las funciones de adquisición descritas en el capítulo anterior y se realizaron pruebas sobre éstas para comprobar su funcionamiento.

Para conformación del portfolio de funciones de adquisición se han seleccionado los principales criterios independientes descritos en 2.4. Se ha excluído la función GP-Hedge ya que se trata en sí de una selección de otras adquisiciones en base a un criterio más complicado.

Por tanto, el conjunto de funciones de adquisición con el que se trabajará será:

$$C = [\alpha_{EI}, \alpha_{PI}, \alpha_{LCB}, \alpha_{AL}] \tag{5.1}$$

donde α_{AL} es una función de adquisición aleatoria. Se ha implementado junto al resto de funciones, haciendo que asigne de manera aleatoria a cada punto un valor según una distribución uniforme $U(0, 1)$, aunque podría haberse hecho de distintas formas (distribución normal, etc). Simula la elección del siguiente punto a evaluar al azar. Introducir esta función en el set es una forma de añadir ruido aleatorio.

En la figura 5.1 se muestra una iteración real del proceso de optimización de un objetivo con la función de adquisición ponderada (*weighted*) con pesos $\frac{5}{8}$ para EI, $\frac{1}{4}$ para PI y $\frac{1}{8}$ para LCB.

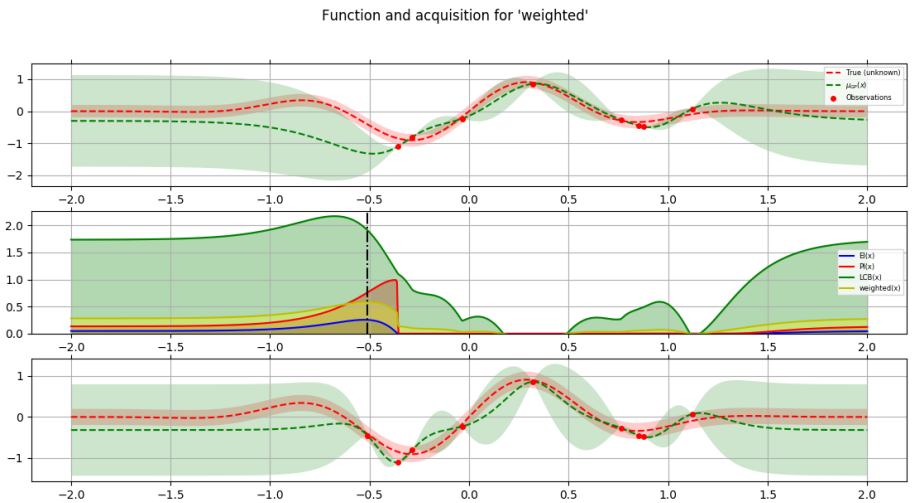


Figura 5.1: Proceso de optimización.

6

Experimentos y resultados

6.1. Descripción y diseño de los experimentos

Una vez comentados los distintos aspectos de la implementación se pasará a comparar los resultados obtenidos en distintos problemas de optimización típicos de la OB (funciones conocidas y con pocas dimensiones), como las funciones de Branin, Rastrigin o Hartmann de 3 dimensiones. Se trata de funciones objetivo tipo *benchmark*, útiles en problemas de optimización por ser continuas, acotadas y relativamente complicadas (tener varios máximos y mínimos locales, etc). Estas en concreto son utilizadas frecuentemente en papers de Optimización Bayesiana. [8] [13] También se añadirá un problema práctico de ajuste de hiperparámetros para ver su comportamiento en un entorno real.

Los experimentos presentados se han basado en el análisis y comparación de la convergencia al mínimo de la Optimización Bayesiana dependiendo de la función de adquisición elegida. Se han ejecutado múltiples veces, con distintas *semillas* aleatorias, para ofrecer resultados más fiables y robustos. Es importante indicar que, con el objetivo de que los distintos criterios sean comparables, se ha lanzado la optimización a igualdad de condiciones los distintos métodos: un mismo punto inicial aleatorio (y sólo uno), mismos parámetros del GP (un kernel Mátern por defecto) y de las funciones de adquisición (por defecto también), mismas semillas para la generación de números aleatorios, etc. Después se ha guardado, para cada iteración, el regret o error cometido respecto al mínimo global en el espacio de búsqueda, que para las funciones benchmark elegidas es conocido. A partir de estos valores se han podido generar gráficos de la convergencia. Las comparaciones se realizarán entre las funciones propuestas, las que componen el set (5.2), y se incluirá GP-Hedge.

Para todas las funciones objetivo se han realizado 100 pasos de optimización excepto para el problema real, que se han hecho 50, ya que no se observaba mejora apreciable, y las ejecuciones resultaban más costosas.

Además, para la función de adquisición ponderada se ha realizado una metaoptimización como la descrita en 4.3, para cada problema, previa a la ejecución de la OB.

A continuación se ofrece una breve descripción de las distintas funciones a optimizar.

Función Branin

Expresión:

$$f(\mathbf{x}) = a(x_2 - bx_1^2 - r)^2 + s(1-t)cos(x_1) + s \tag{6.1}$$

donde $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$ y $t = \frac{1}{8\pi}$.

Normalmente la función se evalúa en el cuadrado $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$.

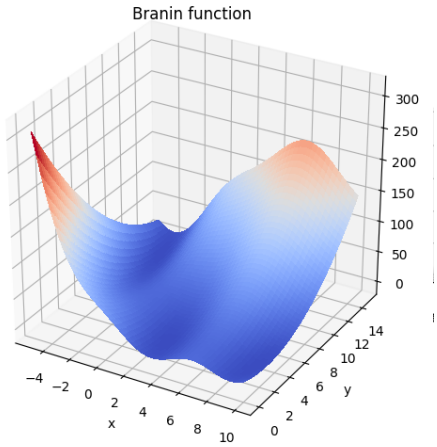


Figura 6.1: Plot de la función Branin.

Función Rastrigin

Expresión:

$$f(\mathbf{x}) = 20 + \sum_{i=1}^{2}[x_i^2 - 10cos(2\pi x_i)] \tag{6.2}$$

con $x_i \in [0, 1]^2$, para $i = 1, 2$.

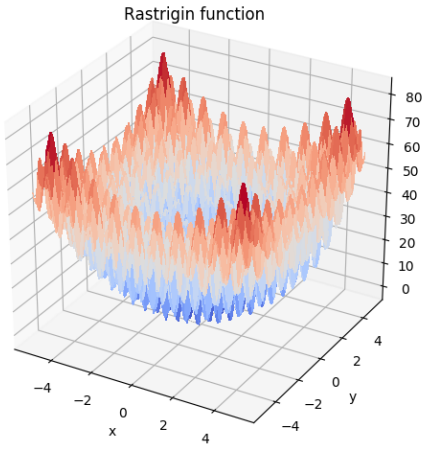


Figura 6.2: Plot de la función Rastrigin.

Función Hartmann de 3 dimensiones

Esta función no es posible representarla dado que el dominio es de 3 dimensiones. Su expresión es:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \beta_i exp(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2) \tag{6.3}$$

donde

$$\beta = (1.0, 1.2, 3.0, 3.2)^T$$

$$A = \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$$

$$P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 38828 \end{pmatrix}$$

evaluada en el hipercubo $x_i \in [0, 1]^3, i = 1, 2, 3$.

Existen versiones de distintas dimensiones. Las más utilizadas son las de 3 y 6 dimensiones.

Problema real de clasificación

La OB realmente no está pensada para las anteriores funciones, de las cuales conocemos la expresión analítica, aunque son una forma de comparar de manera no demasiado costosa distintos criterios. Es por eso que se van a realizar pruebas con un problema real de ajuste de hiperparámetros para un algoritmo de Aprendizaje Automático.

Se ha utilizado el famoso dataset de dígitos escritos manualmente *MNIST* para realizar un entrenamiento y clasificación con el algoritmo *GradientBoostingClassifier* disponible en la librería *scikit-learn*. Los tres hiperparámetros a optimizar han sido el *learning rate* (tasa de aprendizaje), *min_samples_split* (el mínimo número de ejemplos para dividir un nodo interno) y *max_depth* (profundidad máxima de los estimadores individuales). Se ha establecido que el dominio de cada hiperparámetro sea, respectivamente $[0, 1]$, $[0, 1]$ y $[1, 100]$. Como *max_depth* debe ser un entero, se coge el techo del dato tipo *float* correspondiente.

Para el problema real se ha utilizado la misma metodología que con las anteriores funciones. Sin embargo, para cuantificar el regret se ha considerado que, al tratarse de un problema de clasificación, el mínimo error que se puede cometer es 0, con una clasificación perfecta, y por tanto, no es más que la tasa de error (en $[0, 1]$) del clasificador en esa ejecución, con los hiperparámetros correspondientes. Para asegurar que los resultados son independientes de las particiones de datos de entrenamiento y prueba seleccionados, se devuelve la tasa de error media de una validación cruzada con 10 K-Folds y varias repeticiones.

Tabla resumen

A continuación se incluye una tabla a modo de resumen con los aspectos más relevantes de cada uno de los experimentos llevados a cabo:

Función	Branin	Rastrigin	Hartmann-3	Problema real
Dimensiones	2	2	3	3
Espacio de búsqueda	[-5,10]x[0,15]	$[-5.12,5.12]^2$	$[0,1]^3$	$[0, 1]^2$x[1, 100]
Mínimo global	0.397887	0	-3.86278	$\sim$0
Puntos mínimos	($-\pi$, 12.275) (12.275,π) (9.42478,2.475)	(0,0)	(0.114614,0.555649,0.852547)	*Desconocido*

Tabla 6.1: Tabla con detalles de cada función.

6.2. Resultados de los experimentos

En primer lugar, se va a explicar brevemente qué se muestra en las gráficas. Para mantener la coherencia, todas se han generado mediante el mismo script. En él, se han calculado las medias y desviaciones típicas, para cada iteración, del mínimo error absoluto obtenido con respecto al óptimo (regret) por la OB con la función de adquisición correspondiente. Así, se ha representado en cada gráfica el regret en función del número de evaluaciones de la función objetivo.

Para mostrar la convergencia de forma que se aprecien más las diferencias entre los distintos criterios, se han representado en escala logarítmica (base 10), para cada paso de la OB. Además, se ha añadido a cada punto una especie de intervalo de confianza que indica la desviación típica. Por tanto, sea $\bar{x}_t$ la media de los errores mínimos hasta el momento en la iteración t para cierta función de adquisición, y s_t su desviación típica, los valores representados son, $log(\bar{x}_t) \mp log(s_t + 1)$. El $+1$ surge de la necesidad de representar la desviación típica en $[0, +\infty)$.

Debemos hacer una diferenciación entre las funciones *aleatory* y *random*. La primera es la función de adquisición que a cada punto le asigna un ruido aleatorio, es decir, equivalente a selección del siguiente punto a evaluar al azar; y la segunda es la función de selección aleatoria propuesta en la sección 4.2.1.

Función Branin

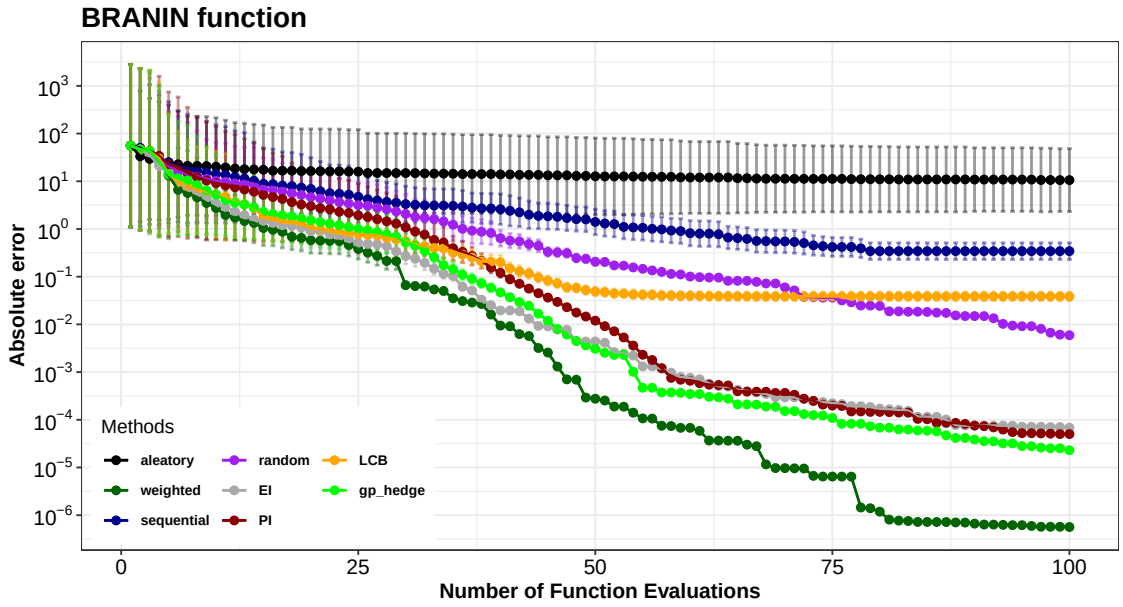


Figura 6.3: Plot de la convergencia para la función Branin.

Vemos la figura 6.3, que representa la convergencia de la OB, con cada función de adquisición, sobre la función Branin. Para comenzar, se debe señalar por qué estos gráficos tienen sentido. La op-

timización comienza cada vez en un punto aleatorio del dominio, por tanto es normal que la desviación típica inicial sea grande, e igual para las distintas adquisiciones. Después, a cada paso, esta desviación va disminuyendo, pues se sigue un criterio concreto, que priorizará puntos parecidos (si no iguales) para distintas ejecuciones de la optimización. La función de adquisición 'aleatory', sin embargo, que no sigue ningún criterio concreto, la ve disminuída en menor medida.

Para este primer experimento los resultados son llamativos: la función de adquisición ponderada ('weighted') supera ampliamente al resto de criterios desde estados muy tempranos de la optimización. Es decir, es un mejor criterio que cualquiera de los que la componen, e incluso que el algoritmo GP-Hedge. Sin embargo, la función secuencial solo se comporta mejor que la búsqueda aleatoria, mientras que la 'random' supera, tras bastantes pasos, a LCB. El resto de criterios se comportan de forma parecida.

Aunque pueda parecer que los resultados de random y secuencial no son demasiado buenos por tener una convergencia lenta, hay que tener en cuenta que sí observamos un comportamiento que ya vaticinábamos al proponerlas: parece que no se 'estancan' tanto como lo hace, por ejemplo, LCB, que no mejora significativamente durante la mitad de la optimización. Esto puede ser útil para determinados problemas donde haya mayor cantidad disponible de evaluaciones de la función objetivo.

Función Rastrigin

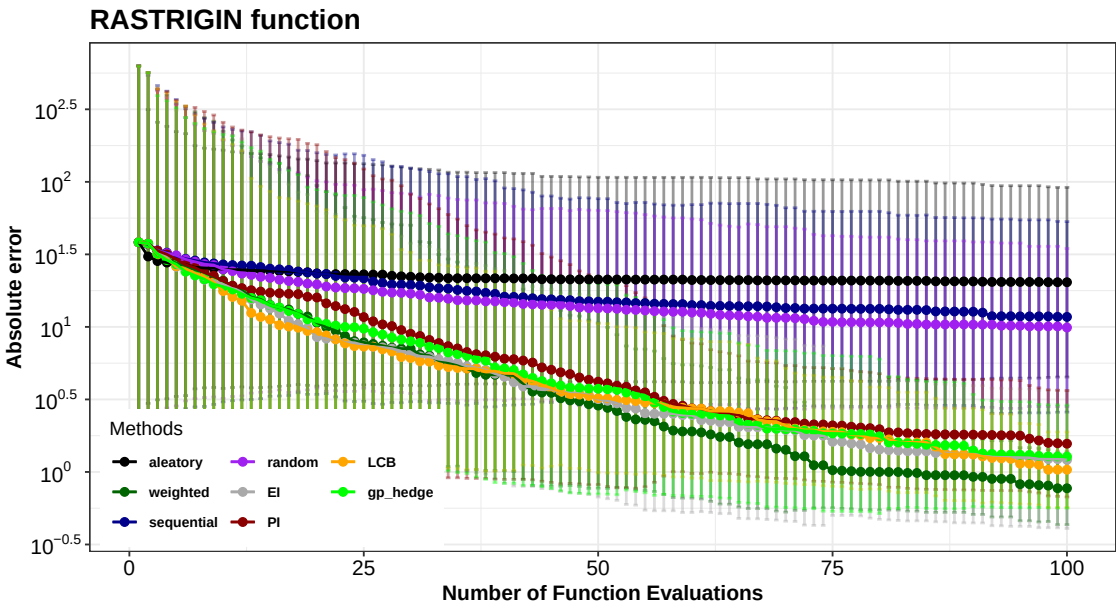


Figura 6.4: Plot de la convergencia para la función Rastrigin.

Esta gráfica tiene peor visibilidad que la anterior debido a que, de forma generalizada, la desviación típica no se reduce demasiado. Observando la representación de la función (6.2) vemos que la función tiene numerosos máximos y mínimos locales cercanos, lo que justifica las amplias diferencias

entre ejecuciones, pues distintos puntos iniciales pueden resultar en mínimos encontrados bastante dispares.

Para esta función objetivo se observan resultados parecidos a los anteriores. Los criterios clásicos se comportan de forma parecida y la función de adquisición ponderada algo mejor a todas ellas, mientras que random y sequential convergen significativamente peor. En este caso parece que el cambio de estrategia (al menos sin criterio basado en las ganancias) queda penalizado.

Función Hartmann de 3 dimensiones

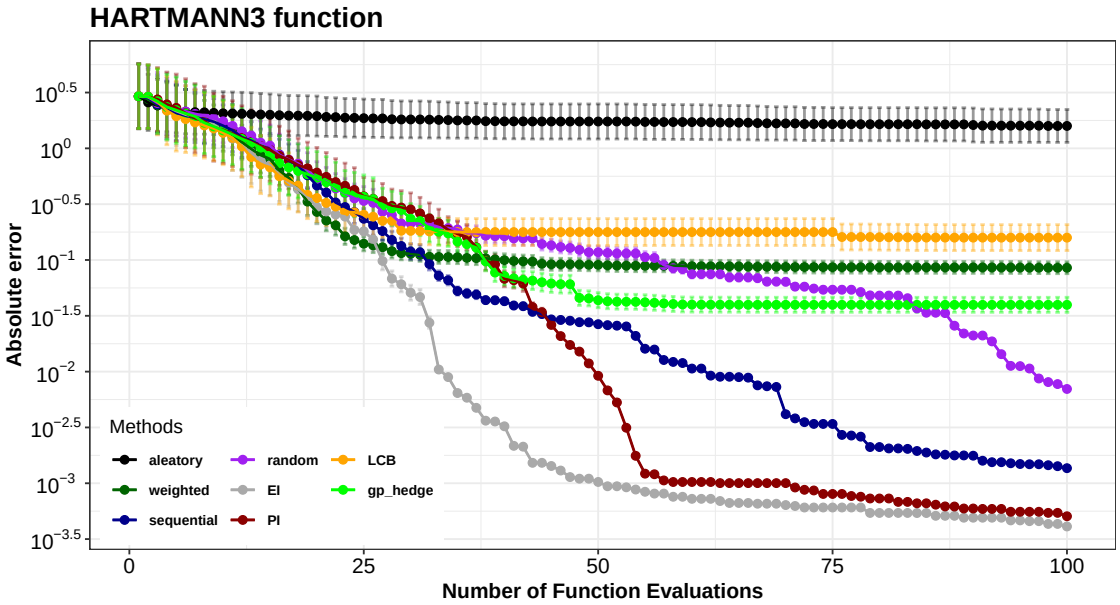


Figura 6.5: Plot de la convergencia para la función Hartmann de 3 dimensiones.

Para la siguiente función cambian las tornas. Vemos en 6.5 que la función de adquisición ponderada, aunque parece buena inicialmente, no mejora visiblemente desde la iteración 25 aproximadamente, y sólo supera a LCB. Por el contrario, los criterios secuencial y random siguen mejorando y superan a GP-Hedge, pero se quedan por detrás de EI y PI, que, aunque con una convergencia distinta, alcanzan un mínimo similar.

Problema real de clasificación

En la gráfica 6.6 pueden observarse diferencias con respecto a las funciones anteriores, pues todas las funciones tienen una convergencia parecida. Dependiendo del dataset, y dado que es un problema de clasificación puede que sea complicado mejorar esa precisión (en torno al 97 % de acierto).

Para este escenario, la función de adquisición aleatoria no se comporta tan mal como en las funciones anteriores, y los métodos clásicos obtienen (ligeramente) peores resultados (y más lentamente) a todos los propuestos. En el siguiente apartado se puede ver el mínimo obtenido con cada uno de los

criterios.

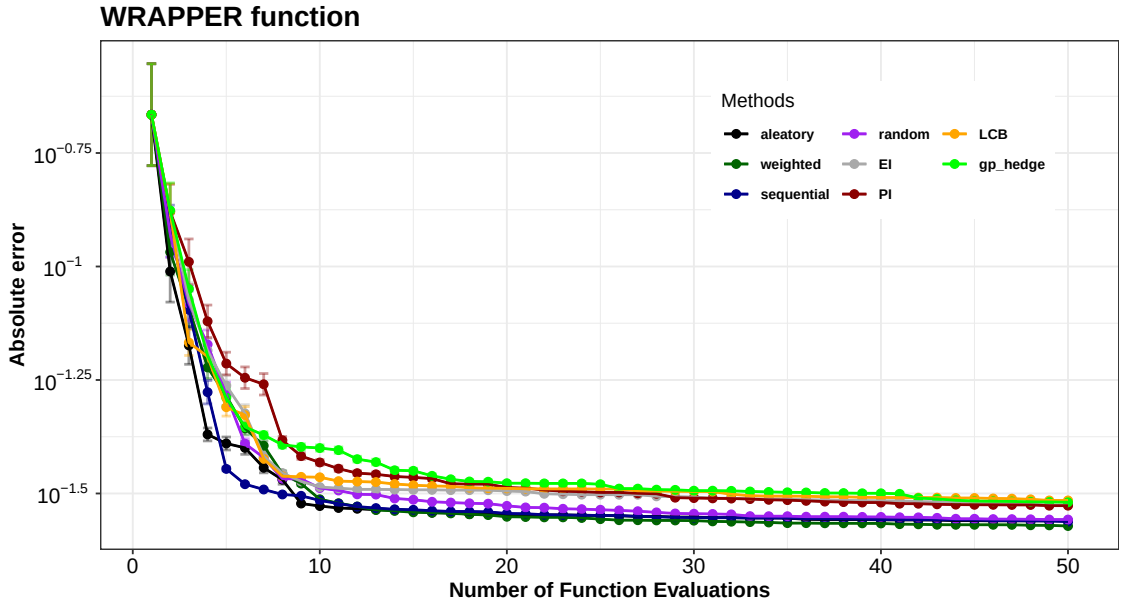


Figura 6.6: Plot de la convergencia para el problema de clasificación real *GradientBoosting* con el dataset *MNIST*.

Tabla de resultados

En la tabla 6.2 se ofrece la media de los errores mínimos absolutos obtenidos para cada función objetivo con cada una de las funciones de adquisición utilizadas.

Función	Branin	Rastrigin	Hartmann-3	Problema real
Aleatoria	10.5841	20.3320	1.5883	0.0275
Ponderada	**5.66x10^{-7}**	**0.7722**	0.0852	**0.0268**
Secuencial	0.3431	11.7271	0.0014	0.0274
Random	0.0059	9.9017	0.0070	0.0277
EI	6.85x10^{-5}	1.2112	**0.0004**	0.0298
PI	5.03x10^{-5}	1.5662	0.0005	0.0297
LCB	0.0386	1.0365	0.1586	0.0305
GP-Hedge	2.30x10^{-5}	1.2764	0.0397	0.0302

Tabla 6.2: Tabla con los resultados de la OB para las distintas funciones objetivo.

7

Conclusiones y trabajo futuro

En este último apartado se van a exponer las conclusiones extraídas a partir de toda la investigación llevada a cabo y una interpretación más genérica de los resultados del apartado de pruebas.

7.1. Objetivos y alcance

Todos los objetivos del proyecto se han alcanzado por completo: se han propuesto nuevos modelos de funciones de adquisición basadas en estrategias portfolio (**O-1.**), se han implementado (**O-2.**), se han realizado e interpretado experimentos sobre las mismas (**O-3.**) y se ha comparado su rendimiento en distintos problemas de optimización (**O-4.**).

Aunque las funciones propuestas sean sencillas, son bastante genéricas y aportan flexibilidad a la hora de incluir distintos criterios de adquisición. Además se ha encontrado un criterio que, aplicando una metaoptimización sencilla previamente (que automatiza el proceso de ajuste de hiperparámetros), no se ha comportado peor (y en algunos casos, lo ha hecho mejor) que cualquier otro en la mayor parte de los problemas (hipótesis **H-1.**): la adquisición ponderada.

No ha habido evidencia que verifique la hipótesis **H-2.**, pero sí se han visto algunas tendencias que podemos extrapolar, como ya hipotetizábamos en **H-3.**.

Se ha respetado y no se ha excedido el alcance fijado. El balance final es positivo y se puede sacar información valiosa que se comentará en los siguientes apartados.

7.2. Impresiones y dificultades

Uno de los mayores problemas encontrados fue el diseño y ejecución de los experimentos. La realización de éstos en varias ejecuciones con distintas semillas para asegurar resultados más fiables y robustos han hecho que los tiempos se dilaten, tardando horas, e incluso días en la mayoría de los casos. Para afrontar esta complicación ha sido de gran ayuda del uso del Centro de Computación Científica de la UAM, donde ha sido posible lanzar numerosos procesos paralelos de optimización para

la obtención de resultados.

En general, el proyecto ha resultado un ejercicio muy valioso al tratarse de mi primer contacto con el mundo de la investigación. El tema de estudio, que prácticamente no conocía previamente, me ha suscitado gran interés, y espero seguir conociendo más del mismo.

7.3. Conclusiones

En primer lugar, conviene indicar que los resultados obtenidos no son uniformes para los distintos problemas y que se encuentra nuevamente la idea del 'No Free Lunch Theorem', dado que las funciones de adquisición que se comportan bien en algunos problemas pagan con un rendimiento más pobre en otros, generalmente.

Sin embargo, sí se han observado algunas tendencias en el apartado de pruebas para los distintos criterios propuestos.

En primer lugar, en la mayoría de problemas, tanto funciones tipo *benchmark* como en el problema práctico, la función de adquisición ponderada obtiene un óptimo mejor (y en ocasiones, antes) que el resto de criterios. Para la función Hartmann no ha sido así.
Esta adquisición no es más que una combinación lineal de las funciones candidatas de un set, por lo que no es trivial que obtenga mejores resultados que cualquiera de ellas individualmente.

También se ha visto empíricamente que las funciones random y secuencial, que van alternando criterios independientemente de su rendimiento, suelen converger más lentamente al no seguir uno de forma constante, pero a cambio no dejan de mejorar en escenarios en los que otras funciones sí. Se trata de una consecuencia del balance entre exploración y explotación.

Por tanto, se han observado comportamientos esperados y, aunque no se ha encontrado una función de adquisición que se comporte mejor en cualquier problema, esto sugiere que el enfoque planteado tiene validez. También se deduce que hay posibilidades de mejora en las funciones de adquisición de OB.

7.4. Trabajo futuro

Los resultados obtenidos son interesantes e invitan a indagar más en esta propuesta, tanto en la prueba de nuevas funciones de adquisición como en la construcción sobre las opciones actuales. A continuación se ofrecen una serie de ideas de proseguimiento o ampliación de este trabajo.

Una primera idea que se sigue naturalmente es el aumento del set de funciones de adquisición y la inclusión de las funciones propuestas dentro de otros conjuntos de funciones. Sería también intere-

sante realizar pruebas con distintas 'instancias', con diferentes parámetros, de una misma función de adquisición.

Otra idea más compleja sería la aplicación de un algoritmo de Aprendizaje Automático que fuera capaz de aprender de las distintas funciones objetivo y el estado del Proceso Gaussiano para determinar qué función o combinación de funciones de adquisición es más adecuado utilizar. Para ello habría que recopilar un amplio dataset con distintas variables como las dimensiones del espacio de búsqueda, las escalas de longitud (*lengthscales*) del proceso, el ratio de convergencia, las funciones de media y covarianzas del GP, el regret obtenido en iteraciones anteriores y con qué función de adquisición, y otras posibles componentes que describan el estado de nuestro proceso de optimización. Esto podría permitir el aprendizaje supervisado de un modelo para poder ser utilizado en una nueva función de adquisición que lo implementara. Gracias a esto se eliminaría de cierta manera la componente heurística.

Bibliografía

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[2] R. Garnett, M. Osborne, and S. Roberts, "Bayesian optimization for sensor set selection," pp. 209–219, 01 2010.

[3] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*, vol. 27, pp. 93–103, Aug 2009.

[4] J. Gonzalez, J. Longworth, D. C. James, and N. D. Lawrence, "Bayesian optimization for synthetic gene design," *arXiv preprint arXiv:1505.01627*, 2015.

[5] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas, "Bayesian optimization in alphago," *arXiv preprint arXiv:1812.06855*, 2018.

[6] T. Weise, "Global optimization algorithms-theory and application," *Self-Published Thomas Weise*, 2009.

[7] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[8] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[9] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT Press Cambridge, MA, 2006.

[10] A. Shah, A. Wilson, and Z. Ghahramani, "Student-t processes as alternatives to gaussian processes," in *Artificial Intelligence and Statistics*, pp. 877–885, 2014.

[11] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter, "Bayesian optimization with robust bayesian neural networks," in *Advances in Neural Information Processing Systems*, pp. 4134–4142, 2016.

[12] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[13] M. D. Hoffman, E. Brochu, and N. de Freitas, "Portfolio allocation for bayesian optimization.," in *UAI*, pp. 327–336, Citeseer, 2011.

[14] J. Wu, M. Poloczek, A. G. Wilson, and P. Frazier, "Bayesian optimization with gradients," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5267–5278, Curran Associates, Inc., 2017.

[15] D. H. Wolpert, W. G. Macready, *et al.*, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[16] T. G. Dietterich *et al.*, “Ensemble learning,” *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[17] M. Molga and C. Smutnicki, “Test functions for optimization needs,” *Test functions for optimization needs*, vol. 101, 2005.

Acrónimos

AA Aprendizaje Automático.

BISP Bayesian Inference in Stochastic Processes.

CCC Centro de Computación Científica.

CDF Cumulative Distribution Function.

EI Expected Improvement.

GP Proceso Gaussiano.

LCB Lower Confidence Bound.

ML Machine Learning.

OB Optimización Bayesiana.

PDF Probability Density Function.

PI Probability of Improvement.

UAM Universidad Autónoma de Madrid.

Apéndices

A

Trabajo auxiliar

En este anexo se incluyen adjuntos un abstract con los avances realizados en este proyecto (A.1), que fue presentado y aceptado para la edición número 11 del Workshop Internacional en Bayesian Inference in Stochastic Processes (BISP) , así como el póster (A.2) que finalmente se presentó y expuso el pasado 12 de Junio de 2019 en la Real Academia de Ciencias de Madrid.

Heuristic Bayesian Optimization

Luis C. Jariego Eduardo C. Garrido-Merchán

February 5, 2019

Bayesian Optimization (BO) methods optimize black-box functions. These black boxes do not have a known expression, are considered to be very expensive, and the observations may be corrupted by noise. BO relies on a probabilistic model, typically a Gaussian Process (GP) that represents the uncertainty of the black box. Iteratively, BO builds an acquisition function (AF) over these GPs which is a criterion that tries to find a good trade-off between exploration and exploitation of the optimization problem. As there exist no best criterion for every possible problem, no free lunch theorem, this work explores heuristics to try to adapt the BO AFs to every possible optimization problem. We are going to present heuristics in order to combine the Probability of Improvement (PI), Expected Improvement (EI) and GP Lower Confidence Bound (GP-LCB) AFs. These heuristics will try to combine these AFs by switching them according to the quality of the evaluations, mixing them by using adaptative or predefined weights in a single AF, choosing them at random or learning by a Machine learning model and a dataset of evaluated GPs which is the best AF for every possible condition of a GP. We expect to obtain improvements in the performance of the different standard criteria present in the SkOpt BO library with the presented heuristics in synthetic problems, benchmark optimization functions and real data scenarios.

1

Figura A.1: Abstract presentado y aceptado en BISP11.

Heuristic Bayesian Optimization

Luis Carlos Jariego Pérez, Eduardo C. Garrido Merchán
Universidad Autónoma de Madrid

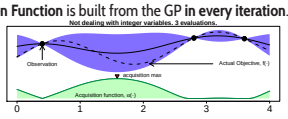
1 - Background

Bayesian Optimization (BO) addresses **black boxes**:

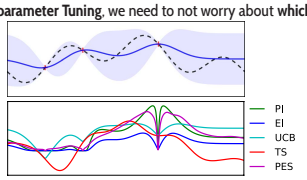
- **Lack of an analytical** expression.
- Very **expensive** to evaluate.
- **Noisy** evaluations.

Automatic Machine Learning tuning of the hyperparameters fits this scenario! But how does BO works?

- BO fits a **Gaussian Process** (GP) to the observations.
- An **Acquisition Function** is built from the GP **in every iteration**.



- But the **Acquisition Function** is a free hyperparameter of BO, it could be a bad choice depending on the problem.
- There is no single **Acquisition Function** that is the best for every problem.
- If we want **Automatic Hyperparameter Tuning**, we need to not worry about **which Acquisition Function to use**.



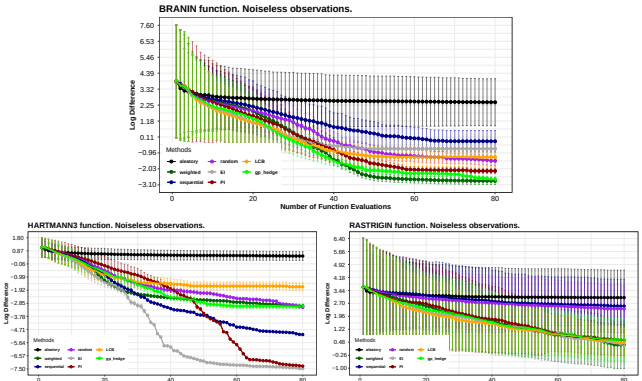
2 - Heuristic Bayesian Optimization

- In this work, we begin to explore the possibilities of **combining Acquisition Functions** in order to build *criteria* that **satisfies the majority of the problems**.
- Formally, if we have a *set* $\mathcal{A}$ of Acquisition Functions, we are going to build criteria that **combines these Acquisition Functions**.
- We hypothesize that *different GP states of an underlying objective function need different Acquisition Functions* in order to discover which is the **optimum of the underlying function**.
- Given the same underlying function which we need to discover the optimum, **different Acquisition Functions will be used in the same BO algorithm!**
- In practice, we have explored **combinations of Standard Acquisition Functions** used in the BO literature:
  - **Probability of Improvement:** $\mathrm{PI}(\mathbf{x}) = \Phi(\frac{f(\mathbf{x}_{best}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})})$.
  - **Expected Improvement:** $\mathrm{EI}(\mathbf{x}) = \sigma(\mathbf{x})(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \phi(\gamma(\mathbf{x})))$.
  - **Lower Confidence Bound:** $\mathrm{LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x})$.
- We propose the **following approaches**, that can be used in an *extended set of Acquisition Functions* like including **PES**, **MES** or any other.
  - **Random criteria:** $Rand(\mathbf{x}) \leftarrow \mathrm{U}\{\mathrm{PI}(\mathbf{x}), \mathrm{EI}(\mathbf{x}), \mathrm{LCB}(\mathbf{x})\}$.
  - **Sequential criteria:** $Seq(\mathbf{x}, n_{iter}) = Cands(\mathbf{x})[n_{iter} mod(n_{cands})]$.
  - **Weighted AF:** $Weigh(\mathbf{x}) = \kappa_{\mathrm{PI}} * \mathrm{PI}(\mathbf{x}) + \kappa_{\mathrm{EI}} * \mathrm{EI}(\mathbf{x}) + \kappa_{\mathrm{LCB}} * \mathrm{LCB}(\mathbf{x})$.
  - **Noised criteria:** $f(\mathbf{x}) = g(\mathbf{x}) + acquisition\_noise * \mathcal{N}(0, 1)$.

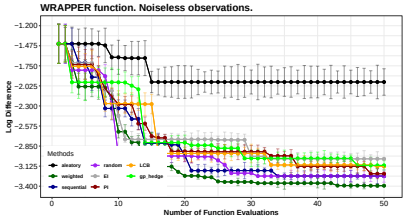
All these approaches are *heuristic* but **explore a space defined by the set** $\mathcal{A}$.

3 - Experiments

The standard Acquisition Function and the proposed ones have been implemented in **SkOpt**.
We test the proposed acquisition functions and compare with GP-Hedge over a set of synthetic and real problems.

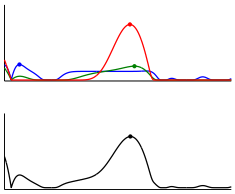


Hyperparameter tuning problem of the *learning rate*, *minimum samples split* and *maximum tree depth* of a **Gradient Boosting Ensemble** classifier on the Digits Dataset.



4 - Visualization and further ideas

Our procedure combines **Acquisition Functions** like this:



- The *Weighted AF criterion* contains a **vector of weights assigning the importance of every Acquisition Function**.
- If, instead of being hardcoded by the user, *these weights are chosen in function of the problem*, the optimization turns to be **automatic**.

We propose to use a **Metaoptimization of the weights** using *Bayesian Optimization over these weights*.

- We define a *search space* of n weights that are associated with n *Acquisition Functions*.
- We execute a **standard Bayesian Optimization procedure** that gives us the **weights that minimize the predicted error**.

5 - Conclusions

- The proposed approaches provide **alternatives** in *Hyperparameter Tuning* problems with respect to the standard **Acquisition Functions**.
- In future work, we will build a dataset of different states of GPs and try to build a model that learns to predict which is the best **Acquisition Function** to use depending on the state of the GP.

Figura A.2: Póster presentado en acsBISP11.

B

Plots

Se incluyen aquí gráficas y resultados que no han tenido cabida en el documento principal y que complementan o extienden su información.

Plot con el resultado de la optimización

En la figura B.1 se muestra el resultado final de un proceso de optimización utilizando Optimización Bayesiana con criterio de adquisición aleatorio sobre una función objetivo de ejemplo con observaciones ruidosas. Se muestra también el Proceso Gaussiano resultante.

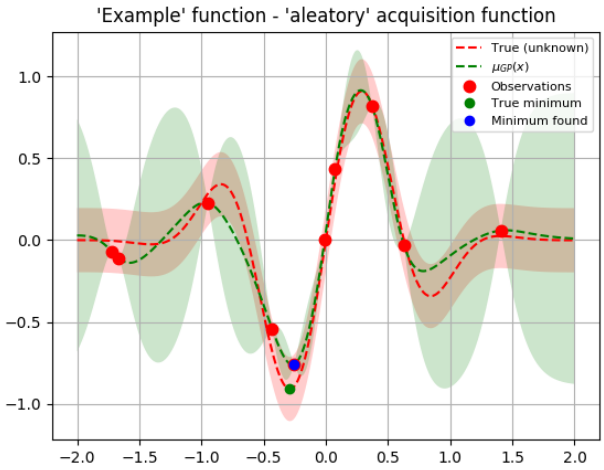


Figura B.1: Plot con el resultado de un proceso de optimización mediante OB con función aleatoria.

En la figura B.2 se muestra el resultado de la misma ejecución con función de adquisición ponderada.

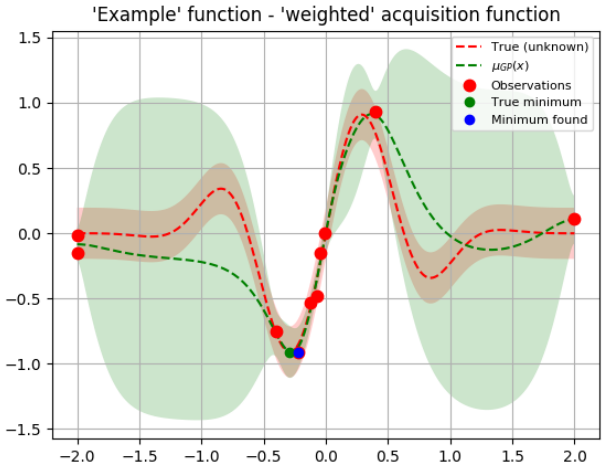


Figura B.2: Plot con el resultado de un proceso de optimización mediante OB con función ponderada.

Función Branin

En B.3 se representan los mínimos de la función Branin y el encontrado por la OB, así como otra perspectiva de la misma.

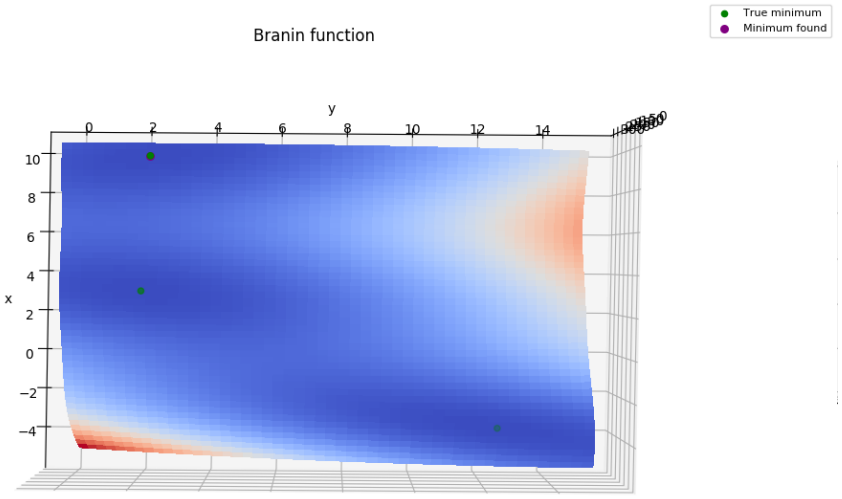


Figura B.3: Mínimos reales de la función Branin y el encontrado por la función de adquisición ponderada en una ejecución.

Función Rastrigin

B.4 muestra lo mismo, de forma análoga, para la función Rastrigin.

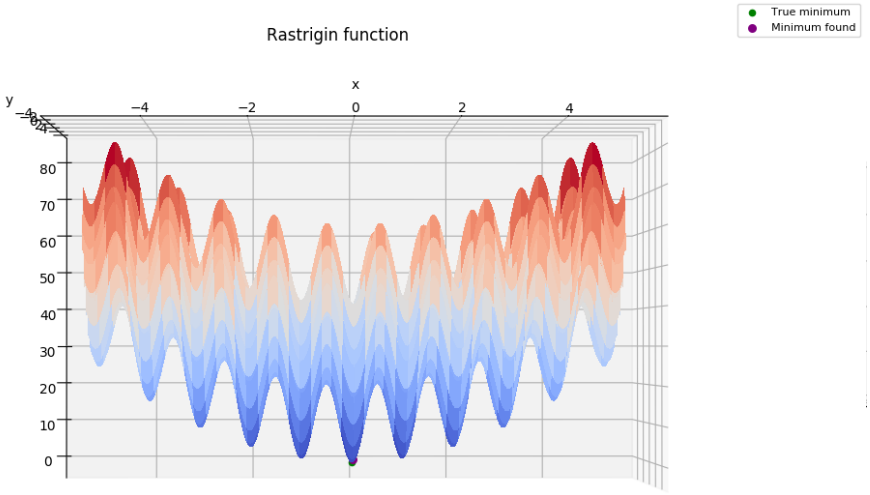


Figura B.4: Mínimo real y el encontrado para Rastrigin por la función de adquisición ponderada.

Función Schaffer nº2

B.5 muestra la convergencia de una función típica en optimización de entre otras que se han probado. No se ha añadido al documento principal por no ser útil para la mayoría de los criterios. Se observa un buen rendimiento para la función de adquisición ponderada, pese a que en este caso impera el criterio aleatorio.

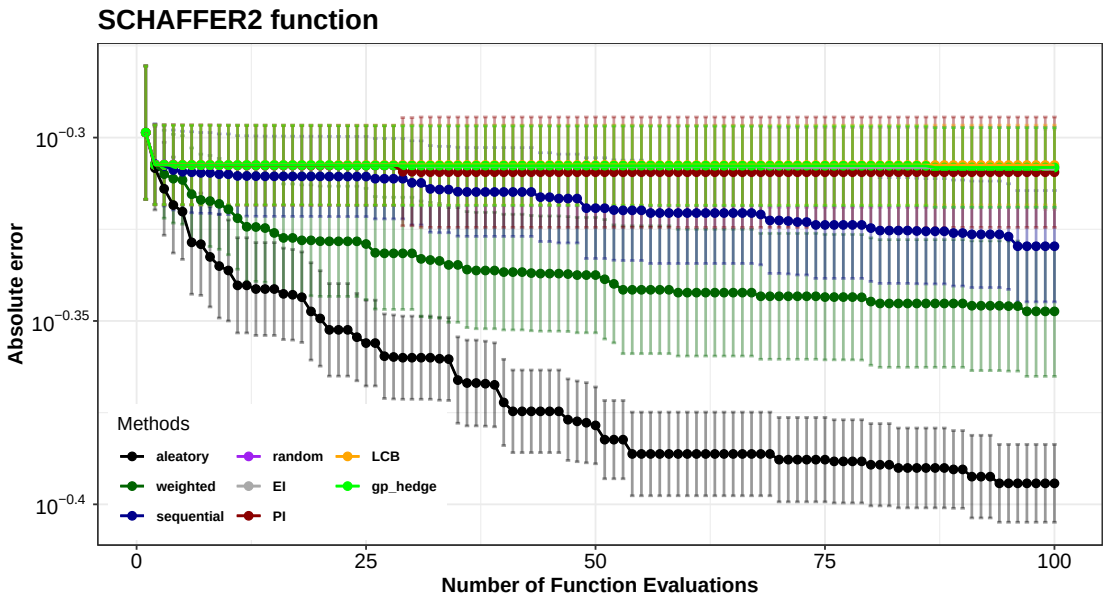


Figura B.5: Plot de la convergencia para la función Schaffer nº2.